An Overview of *Meros*

Trilinos User's Group

Wednesday, November 2, 2005

Victoria Howle
Computational Sciences and Mathematics Research Department (8962)

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company,
for the United States Department of Energy under contract DE-AC04-94AL85000.

Outline

- What is Meros?
- Motivation & background
  - Incompressible Navier–Stokes
  - Block preconditioners
- Some preconditioners being developed in Meros
- A few results from these methods
- Code example: user level
- Code example: inside Meros
- Release plans, etc.
- References

What is Meros?

- Segregated preconditioner package in Trilinos
- Scalable block preconditioning for problems that couple simultaneous solution variables
- Initial focus is on (incompressible) Navier-Stokes
- Release version in progress
  - Updating (from old TSF) to Thyra interface
  - Plan to release next Fall ‘06
- Team
  - Ray Tuminaro
    1414, Computational Mathematics & Algorithms
  - Robert Shuttleworth
    Univ. of Maryland, Summer Student Intern 2003, 2004, 2005
- Other collaborators
  - Howard Elman, University of Maryland
  - Jacob Schroder, University of Illinois, Summer Intern 2005
  - John Shadid, Sandia, NM
  - David Silvester, Manchester Univerity

Where is Meros in *The Big Picture*

Analyst or Designer

Optimization Code
(e.g., APPSPACK, MOOCHO, Opt++, Split)

Simulation Code
(e.g., MPSalsa, Sierra, Sundance)

Nonlinear Solver
(e.g., NOX)

Linear Solver
(e.g., AZTECOO, Belos)
Preconditioner
(e.g., IFPACK, *Meros*, ML)

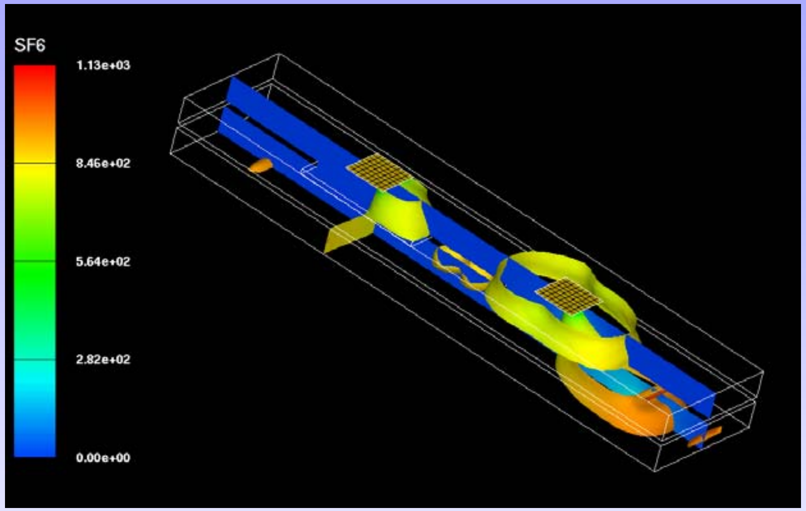
Linear Algebra Kernel
(e.g., Epetra)

- The speed, scalability, and robustness of an application can be heavily dependent on the speed, scalability, and robustness of the linear solvers
- Linear algebra often accounts for >80% of the computational time in many applications
- Iterative linear solvers are essential in ASC-scale problems
- Preconditioning is the key to iterative solver performance

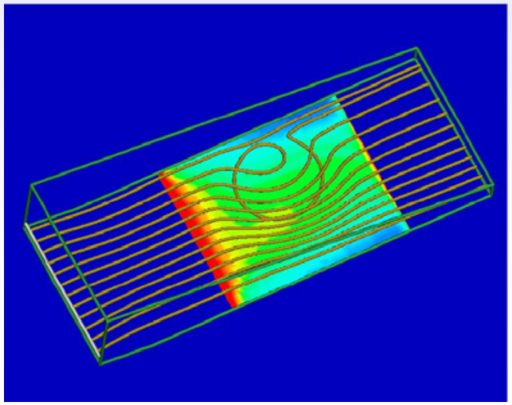
Incompressible Navier–Stokes

- Examples of incompressible flow problems
  - Airflow in an airport; e.g., transport of an airborne toxin
  - Chemical Vapor Deposition
- Goal: efficient and robust solution of steady and transient chemically reacting flow applications
- Current testbed application: MPSalsa
- Early user: Sundance
- Related Sandia applications:
  - Charon
  - ARIA
  - Fuego
  - …

Airport source detection problem



CVD Reactor



CSMR
Computational Sciences and Mathematics Research

Sandia National Laboratories

Incompressible Navier–Stokes

$$\begin{aligned} \alpha \mathbf{u}_t - \nu \nabla^2 \mathbf{u} + (\mathbf{u} \cdot \mathsf{grad})\,\mathbf{u} + \mathsf{grad}\, p &= \mathbf{f} \\ -\mathsf{div}\, \mathbf{u} &= 0 \end{aligned}$$

$$\begin{pmatrix} F & G \\ G^T & -C \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ p \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ \mathbf{g} \end{pmatrix}$$

- $\alpha = 0 \Rightarrow$ steady state, $\alpha = 1 \Rightarrow$ transient
- (2,2)-block = 0 (unstabilized) or
= C (stabilized)
- Incompressibility constraint $\Rightarrow$ difficult for linear solvers
- Chemically reactive flow $\Rightarrow$ multiphysics; even harder
- Indefinite, strongly coupled, nonlinear, nonsymmetric systems

Block preconditioners

- Want the scalability of multigrid (*mesh-independence*)
- Difficult to apply multigrid to the whole system
- Solution:
  - Segregate blocks and apply multigrid separately to subproblems
- Consider the following class of preconditioners:

$$\mathcal{M}^{-1} = \begin{pmatrix} F & G \\ 0 & -\tilde{S} \end{pmatrix}^{-1}$$

$$= \begin{pmatrix} F^{-1} & \\ 0 & I \end{pmatrix} \begin{pmatrix} I & G \\ & I \end{pmatrix} \begin{pmatrix} I & \\ & -\tilde{S}^{-1} \end{pmatrix}$$

- $\mathcal{M}$ is an optimal (right) preconditioner when $\tilde{S}$ is the Schur complement, $S = G^T F^{-1} G$

(Assuming C = 0)

Choosing $\tilde{S}$ (Kay & Loghin, F_p)

- Key is choosing a good Schur complement approximation $\tilde{S}$ to $S = G^T F^{-1} G$
- Motivation: move F^{-1} so that it does not appear between G^T and G
- Suppose we have an F_p such that $FG = GF_p$

Then $$GF_p^{-1} = F^{-1}G$$

And $$S = G^T G F_p^{-1} \Rightarrow S^{-1} = F_p (G^T G)^{-1}$$

Giving $$\mathcal{M}^{-1} = \begin{pmatrix} F^{-1} & \\ 0 & I \end{pmatrix} \begin{pmatrix} I & G \\ & I \end{pmatrix} \begin{pmatrix} I & \\ & -F_p(G^T G)^{-1} \end{pmatrix}$$

(Kay, Loghin, & Wathen; Silvester, Elman, Kay & Wathen)

$$\tilde{S}^{-1} = F_p A_p^{-1}$$ (A_p is pressure Poisson)

Other Choices for $\tilde{S}$

- Kay & Loghin F_p method works well, but…
  - F_p is not a standard operator for apps (pressure convection–diffusion)
  - Can be difficult for many applications to provide
  - Even if they *can* provide it, they don't really want to
- Other options for $\tilde{S}$:
  Algebraic pressure convection–diffusion methods:
  - Sparse Approximate Commutator (SPAC)
  - Least Squares Commutator (LSC)
- Algebraically determine an operator F_p such that

$$GF_p \approx FG$$

$$\min_{F_p} \|GF_p - FG\|_F^2$$

CSMR
Computational Sciences and Mathematics Research

Sandia National Laboratories

Algebraic Commutators

$$\min_{F_p} \|GF_p - FG\|_F^2$$

- Build F_p column by column via ideas similar to sparse approximate inverses (e.g., Grote & Huckle) $\Rightarrow$ Sparse Approximate Commutators (SPAC)
  - $\tilde{S}^{-1} = F_p(G^T G)^{-1}$
  - F_p is no longer a pressure convection-diffusion operator
- Minimize via normal equations $\Rightarrow$ Least Squares Commutators (LSC)
  - $\tilde{S}^{-1} = F_p(G^T G)^{-1}$
  - $F_p = (\tilde{G}^T \tilde{G})^{-1} \tilde{G}^T \tilde{F} \tilde{G}$
  - $\tilde{S}^{-1} = (\tilde{G}^T \tilde{G})^{-1} \tilde{G}^T \tilde{F} \tilde{G} (\tilde{G}^T \tilde{G})^{-1}$
- The tilde's are hiding an issue of algebraic vs. differential commuting

$$\tilde{G} = M_d^{-\frac{1}{2}} G, \qquad \tilde{F} = M_d^{-\frac{1}{2}} F M_d^{-\frac{1}{2}}$$

CSMR
Computational Sciences and Mathematics Research

Sandia National Laboratories

Stabilized LSC (C ≠ 0)

- Certain discretizations require stabilization
- Stabilization term C

$$S = G^T F^{-1} G + C$$

- For certain discretizations ($G^T G$) is unstable
  - Blows up on high frequencies
  - C built to stabilize $(G^T F^{-1} G)$
- Preconditioner $\tilde{S}^{-1} = (\tilde{G}^T \tilde{G})^{-1} \tilde{G}^T \tilde{F} \tilde{G} (\tilde{G}^T \tilde{G})^{-1}$ also needs stabilization
  - In 3 places
  - Use C for preconditioner stabilization, too

$$\tilde{S}_\alpha^{-1} = W^{-1} G^T F G W^{-1} + \alpha D^{-1} \qquad W = (G^T G + \gamma C)$$

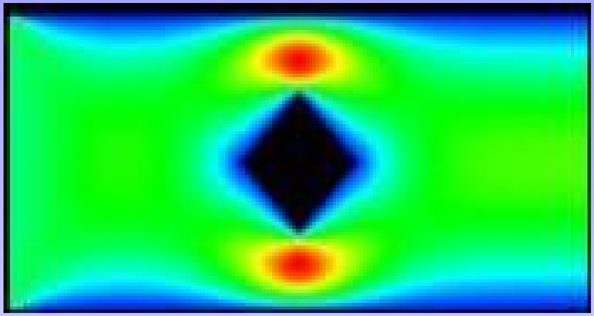
$$\tilde{S}_\sigma^{-1} = W^{-1} (G^T F G + \sigma C) W^{-1}$$

CSMR Computational Sciences and Mathematics Research

Sandia National Laboratories

F_p vs. DD results: Flow over a diamond in MPSalsa

- Linear solve timings
- Steady state (harder than transient for linear algebra)
- Parallel (on Sandia's ICC)
- Re = 25



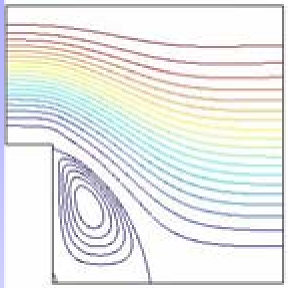
Unknowns (Procs)	DD (seconds)	F_p (seconds)
62K (1)	199	368
256K (4)	1583	736
1M (16)	7632	1428
4M (64)	failed	5532

- Using development version of Meros hooked into MPSalsa through NOX

(Matlab) Results: F_p, LSC, and SPAC

- Linear iterations for backward facing step problem on underlying 64x192 grid, Q_2-Q_1 (stable) discretization.

Re	F_p	LSC	SPAC
10	30	19	23
100	42	21	30
200	47	22	41



- Linear iterations for backward facing step problem on underlying 128x384 grid, Q_2-Q_1 (stable) discretization.

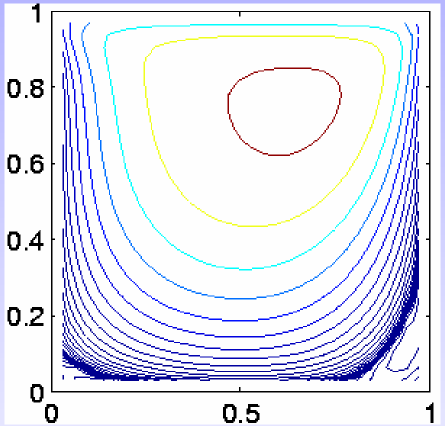
Re	F_p	LSC	SPAC
10	33	23	32
100	58	29	39
200	63	29	60

- Results from *ifiss*
  - Academic software package that incorporates our new methods and a few other methods (Elman, Silvester, Ramage)

(Matlab) Results: F_p, LSC, Stabilized LSC

- Linear iterations for lid driven cavity problem on 32x32 grid, Q_1-Q_1 (needs stabilization) discretization.

Re	F_p	LSC	Stabilized LSC
100	27	151	16
500	57	197	29
1000	80	228	44
5000	130	320	83



- Results from *ifiss*

Transition promising academic methods into methods for ASC applications

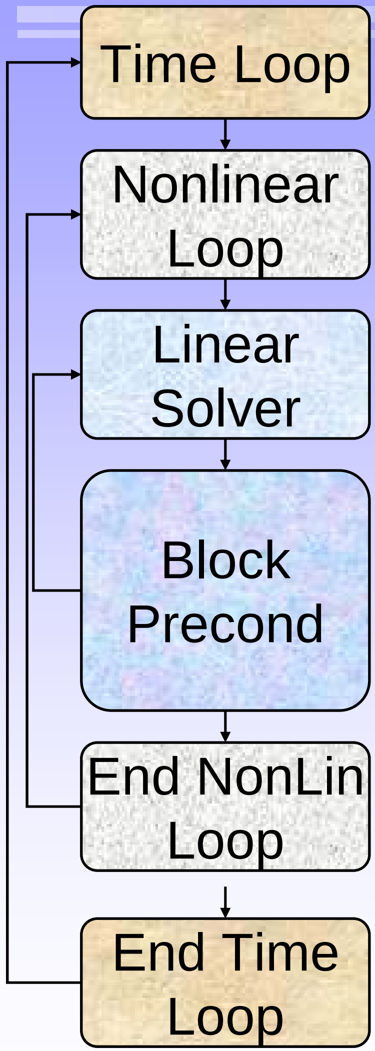
- Promising methods have been developed
- We have extended these methods mathematically to suit more realistic needs
  - Removing need for nonstandard operators
  - Stabilization
- Currently, software for these methods is mostly in academic (Matlab) codes
- Now need to develop software to make them available to more real-world apps through Trilinos

Meros

- Initial focus is on preconditioners for Navier-Stokes
- A number of solvers are being incorporated:
  - Pressure convection-diffusion preconditioners (today's focus)
    - F_p (Kay & Loghin)
    - LSC (and stabilized LSC)
    - (SPAC?)
  - Pressure-projection methods
    E.g., SIMPLE (SIMPLEC, SIMPLER, etc.)

$$
\begin{aligned}
P^{-1} &= \begin{bmatrix} I & -D^{-1}G \\ 0 & I \end{bmatrix} \begin{bmatrix} F & 0 \\ 0 & \tilde{S} \end{bmatrix}^{-1} \\
\tilde{S}^{-1} &= G^T D^{-1} G \\
D &= diag(F)
\end{aligned}
$$

Trilinos packages in an MPSalsa example



Component	Methods	Package
MPSalsa	Finite Element	Epetra
Nonlinear Solver	Newton-Krylov Methods	NOX
Linear Solver	GMRESR	Aztec00 (Epetra, Thyra)
block preconditioner	$\begin{bmatrix} F^{-1} & \\ & \tilde{S}^{-1} \end{bmatrix}$	**Meros** (Thyra)
	F^{-1} : GMRES/AMG **$\hat{S}^{-1}$: CG/AMG**	Aztec00, ML Epetra

Meros & Trilinos

- Meros is a package within Trilinos
- Meros is also a user of many other Trilinos packages
- Depends on:
  - Thyra
  - Teuchos
  - (Epetra)
- Currently uses:
  - AztecOO
  - IFPACK
  - ML
- Could use:
  - Belos
  - Amesos
  - …

Example preconditioner: First set up abstract solvers for inner solves

```
// WARNING: Assuming TSF-style handles and assuming I have typedeffed
    // to hide the Templating
// WARNING: Examples include functionality that is not yet available in Thyra

// Meros builds a PreconditionerFactory so we can pass it to an abstract linear
    solver

// E.g., K & L preconditioner needs the saddlepoint matrix A, plus F_p and A_p,
// and choices of solvers for F and A_p

// Inner F solver options:
Teuchos::ParameterList  FParams;
FParams.set("Solver", "GMRES");
FParams.set("Preconditioner", "ML");
FParams.set("Max Iters", 200);
FParams.set("Tolerance", 1.0e-8);   // etc
LinearSolver FSolver = new AztecSolver(FParams);

// Inner Ap solver options:
ApParams.set("Solver", "PCG");
ApParams.set("Preconditioner", "ML"); // etc
LinearSolver ApSolver = new AztecSolver(ApParams);
```

Next set up Schur complement approx. and build the preconditioner

```
// Set up Schur complement approx factory (with solvers if necessary)
SchurFactory sfac = new KayLoghinSchurFactory(ApSolver);

// Build preconditioner factory with these choices
PreconditionerFactory pfac = new KayLoghinFactory(outerMaxIters,
                                        outerTol, FSolver, sfac, …)
// Group operators that are needed by preconditioner
OperatorSource opSrc = new KayLoghinOperatorSource(saddleA, Fp, Ap);

// Use preconditioner factory directly in an abstract solver
outerParams.set("Solver","GMRESR"); // etc
LinearSolver solver = new AztecSolver(outerParams);
SolverState solverstate = solver.solve(pfac, opSrc, rhs, soln);
```

Example (cont.)

```
// Get Thyra Preconditioner from factory for a particular set of ops
Preconditioner Pinv =  pfac.createPreconditioner(opSrc);

// Get Thyra LinearOpWithSolve to use precond op more directly
LinearOperator Minv = Pinv.right();
outerParams.set("Solver","GMRESR");
LinearSolver solver = new AztecSolver(outerParams);
SolverState solverstate = solver.solve(A*Minv, rhs, intermediateSoln);
soln = Minv * intermediateSoln;


// Simple constructors will make intelligent choices of defaults:
PreconditionerFactory pfac = new KayLoghinFactory(maxIters, Tol);

// Still need the appropriate operators for the chosen method
// (some can be built algebraically by default if not given, e.g., SPAC)
OperatorSource opSrc = new OperatorSource(A, Fp, Ap);
```

Inside createPreconditioner()

```
// Build the preconditioner given 2x2 block matrix (etc.)
Preconditioner KayLoghinFactory::createPreconditioner(…)
{

    // Get F, G, G^T blocks from the block operators
    LinearOperator F = A.getBlock(1,1);
    LinearOperator G = A.getBlock(1,2);
    LinearOperator Gt = A.getBlock(2,1);
    // LinearOperators Ap and Fp built here or gotten from OpSrc

    // Set up F solve (given solver and parameters or build with defaults)
    LinearOpWithSolve Finv = F.inverse(FSolver);
```

$$\mathcal{M}^{-1} = \begin{pmatrix} F^{-1} & \\ 0 & I \end{pmatrix} \begin{pmatrix} I & G \\ & I \end{pmatrix} \begin{pmatrix} I & \\ & \tilde{S}^{-1} \end{pmatrix}$$

createPreconditioner() (cont.)

```
// Setup Schur complement approximation and solver
// (given by user or build using defaults)
LinearOpWithSolve Apinv = Ap.inverse(ApSolver);
LinearOperator Sinv = -Fp * Apinv;

// Or if we were building an LSC preconditioner
LinearOperator GtG = Gt * G;
LinearOpWithSolve GtGinv = Ap.inverse(ApSolver);
LinearOperator Sinv = -GtGinv * Gt * F * G * GtGinv;
```

$$\mathcal{M}^{-1} = \begin{pmatrix} F^{-1} & \\ 0 & I \end{pmatrix} \begin{pmatrix} I & G \\ & I \end{pmatrix} \begin{pmatrix} I & \\ & \tilde{S}^{-1} \end{pmatrix}$$

createPreconditioner() (cont.)

```
LinearOperator Iv = IdentityOperator(F.domain()); // velocity space
LinearOperator Ip = IdentityOperator(G.domain()); // pressure space

// Domain and range of A are Thyra product spaces, velocity x pressure
LinearOperator P1 = new BlockLinearOp(A.domain(),A.range());
LinearOperator P2 = new BlockLinearOp(A.domain(),A.range());
LinearOperator P3 = new BlockLinearOp(A.domain(),A.range());

P1.setBlock(1,1,Finv);
P1.setBlock(2,2,Ip);
P2.setBlock(1,1,Iv);
P2.setBlock(2,2,Ip);
P2.setBlock(1,2,G);
P3.setBlock(1,1,Iv);
P3.setBlock(2,2,Sinv);

return new GenericRightPreconditioner(P1*P2*P3);
}
```

$$\mathcal{M}^{-1} = \begin{pmatrix} F^{-1} & \\ 0 & I \end{pmatrix} \begin{pmatrix} I & G \\ & I \end{pmatrix} \begin{pmatrix} I & \\ & \tilde{S}^{-1} \end{pmatrix}$$

CSMR Computational Sciences and Mathematics Research

Sandia National Laboratories

Plans & Info

- Planning to release *Meros 1.0* in Fall ’06 (with closest major Trilinos release)

- Initial block preconditioner selection should include:
  - Pressure Convection-Diffusion
    - Kay & Loghin (F_p)
    - Least Squares Commutator (LSC)
    - SPAC?
  - Pressure Projection
    - SIMPLE
    - SIMPLEC, SIMPLER?

- Web page: software.sandia.gov/Trilinos/packages/meros/index.html
- Mailing lists: Meros-Announce, Meros-Users, etc.
- vehowle@sandia.gov

References

- Elman, Silvester, and Wathen, *Performance and analysis of saddle point preconditioners for the discrete steady-state Navier-Stokes equations*, Numer. Math., 90 (2002), pp. 665-688.
- Kay, Loghin, and Wathen, *A preconditioner for the steady-state Navier-Stokes equations*, SIAM J. Sci. Comput., 2002.
- Elman, H., Shadid, and Tuminaro, *A Parallel Block Multi-level Preconditioner for the 3D Incompressible Navier-Stokes Equations*, J. Comput. Phys, Vol. 187, pp. 504-523, May 2003.
- Elman, H., Shadid, Shuttleworth, and Tuminaro, *Block Preconditioners Based on Approximate Commutators*, to appear in SIAM J. Sci. Comput., Copper Mountain Special Issue, 2005.
- Elman, H., Shadid, and Tuminaro, *Least Squares Preconditioners for Stabilized Discretizations of the Navier-Stokes Equations*, in progress.